Listing of Claims:

1. (Withdrawn) A deferred graphics pipeline processor comprising:

a texture unit and a texture memory associated with said texture unit; said texture unit applying to texture maps stored in said texture memory, to pixel fragments; said textures being MIP-mapped and comprising a series of texture maps at different levels of detail, each map representing the appearance of comprising a series of texture maps at different levels of detail, each map representing the appearance of texture at a given distance from an eye point; said texture unit performing tri-linear interpolation from said texture maps to produce a texture value for a given pixel fragment that approximates the correct level of detail; said texture memory having texture data stored and accessed in a manner which reduces memory access conflicts and thus improves throughput of said texture unit.

2. (Previously Presented) An address reordering device for use in a memory system, the device comprising:

(1) a memory control block receiving dispatched addresses and sequentially performing read operations on the memory system using the dispatched addresses;

(2) a first level reorder queue storing a plurality of current addresses, each of the current addresses being equal to one of the dispatched addresses;

(3) a conflict queue storing stalled addresses, each of the stalled addresses being an address into texture memory that has its corresponding read operation postponed due to an address conflict;

(4) an in-order tag queue storing first tag information, each piece of first tag information corresponding to an address in the first level reorder queue;

(5) an out-of-order tag queue storing second tag information, each piece of second tag information corresponding to an address in the conflict queue;

(6) a conflict detection block comprising:

(6a) logic receiving a new address into texture memory, the new address being part of the sequence of addresses being received in a specific order;

(6b) logic detecting a memory conflict between the new address and any of the plurality of current addresses;

(6c) logic dispatching the new address to the memory control block if the conflict was not detected so as to make the new address into one of the dispatched addresses;

(6d) logic writing the new address into the first level reorder queue if the conflict is not detected so as to make the new address into one of the current addresses;

(6e) logic writing the new address into the conflict queue if the conflict is detected so as to make the new address into one of the stalled addresses;

(6f) logic writing new tag information corresponding to the new address into the in-order tag queue if the conflict is not detected;

(6g) logic writing the new tag information corresponding to the new address into the out-of-order tag queue if the conflict is detected; and

(6h) logic determining when the stalled addresses are dispatched to the memory control block; and

(7) logic reassembling data read from the memory system into the specific order, the reassembling being done according to the first tag information and the second tag information.

3. (Previously Presented) An address reordering method for use in a memory system, the method comprising the steps:

maintaining a list of current addresses, each of the current addresses being an address into a memory system that has been dispatched to the texture memory as part of a memory read operation that has not yet completed;

maintaining a list of stalled addresses, each of the stalled addresses being an address into the memory system that has its corresponding read operation postponed due to an address conflict;

maintaining a list of first tag information, each piece of first tag information corresponding to an address in the list of current addresses;

maintaining a list of second tag information, each piece of second tag information corresponding to an address in the list of stalled addresses;

receiving a new address into texture memory, the new address being part of a sequence of addresses being received in a specific order;

detecting the presence of a memory conflict between the new address and any of the current addresses;

if the conflict is not detected, dispatching the new address to perform a read operation from the texture memory;

if the conflict is not detected, adding the new address to the list of current;

if the conflict is detected, adding the new address to the list of stalled addresses;

if the conflict is not detected, adding the new tag information corresponding to the new address to the list of first tag information;

if the conflict is detected, adding the new tag information corresponding to the new address to the list of second tag information;

determining when the stalled addresses are dispatched to the memory control block; and

reassembling data read from the memory system into the specific order, the reassembling being done according to the first tag information and the second tag information.

4. (Previously Presented) The method of claim 3, the method further comprising the step:

performing a programmable mapping function of the bits within the dispatched address to (1) device bits selecting one or more memory devices from a plurality of memory devices; and (2) bank bits selecting a memory bank within the selected memory device.

5. (withdrawn) An address reordering method for use in a memory system, wherein the memory system includes a texture memory coupled to a texel prefetch buffer, the method comprising:

receiving a plurality of requests for information not contained in the texel prefetch buffer, each of the plurality of requests including at least one address;

reordering the plurality of requests according to the addresses included in each request;

coupling the reordered plurality of requests to the texture memory.

6. (withdrawn) An address reordering method according to claim 5, the act of reordering the plurality of requests comprising placing a first received request of the plurality of requests in a first-level reorder queue and determining if a second received request conflicts with any requests in the first-level reorder queue.

7. (withdrawn) An address reordering method according to claim 6, further comprising, if the second received request conflicts with the first received request, placing the second received request in a conflict queue.

8. (withdrawn) An address reordering method according to claim 6, further comprising, if the second received request does not conflict with the first received request, placing the second received request in the first-level reorder queue.

9. (withdrawn) An address reordering method according to claim 5, further comprising: prioritizing the plurality of requests according to type of request, wherein the type of request may include a texture look up request, a copy texture request, or a read texture request.

10. (withdrawn) An address reordering method according to claim 5, wherein the texture memory comprises at least one content addressable memory.

11. (withdrawn) An address reordering method according to claim 5, wherein the texture memory comprises a plurality of content-addressable memories.

12. (withdrawn) An address reordering method according to claim 10, wherein the content addressable memory stores data and tag information separately, and outputs a plurality of indices.

13. (withdrawn) An address reordering method according to claim 5, further comprising:
accessing a texel, according to at least one of the received requests; and
saving the texel for a predetermined period of time after the received request.

14. (withdrawn) An address reordering method according to claim 13, wherein the act of saving the texel comprises storing the texel in the texture memory.

15. (withdrawn) An address reordering method according to claim 13, further comprising receiving a second request for the texel and reusing the stored texel, responsive to the second request.

16. (Previously Presented) An address reordering device for use in a memory system, the device comprising:

a memory control block receiving dispatched addresses and sequentially performing read operations on the memory system using the dispatched addresses;

a first level reorder queue storing a plurality of current addresses, each of the current addresses being equal to one of the dispatched addresses;

a conflict queue storing stalled addresses, each of the stalled addresses being an address into texture memory that has its corresponding read operation postponed due to an address conflict;

an in-order tag queue storing first tag information, each piece of first tag information corresponding to an address in the first level reorder queue;

an out-of-order tag queue storing second tag information, each piece of second tag information corresponding to an address in the conflict queue;

a conflict detection and control logic unit; and

a reassembly logic unit reassembling data read from the memory system into the specific order, the reassembling being done according to the first tag information and the second tag information.

17. (new) An address reordering method for use in a memory system comprising:

receiving a plurality of requests for information not contained in a buffer of the memory system, each of the plurality of requests including at least one address;

reordering the plurality of requests according to the addresses included in each request;

coupling the reordered plurality of requests to a texture memory coupled to the buffer.

18. (new) An address reordering method according to claim 17, the act of reordering the plurality of requests comprising placing a first received request of the plurality of requests in a first-level reorder queue and determining if a second received request conflicts with any requests in the first-level reorder queue.

19. (new) An address reordering method according to claim 18, further comprising, if the second received request conflicts with the first received request, placing the second received request in a conflict queue.

20. (new) An address reordering method according to claim 18, further comprising, if the second received request does not conflict with the first received request, placing the second received request in the first-level reorder queue.

21. (new) An address reordering method according to claim 17, further comprising:

prioritizing the plurality of requests according to type of request, wherein the type of request may include a texture look up request, a copy texture request, or a read texture request.

22. (new) An address reordering method according to claim 17, wherein the texture memory comprises at least one content addressable memory.

23. (new) An address reordering method according to claim 17, wherein the texture memory comprises a plurality of content-addressable memories.

24. (new) An address reordering method according to claim 22, wherein the content addressable memory stores data and tag information separately, and outputs a plurality of indices.

25. (new) An address reordering method according to claim 17, further comprising:
accessing a texel, according to at least one of the received requests; and
saving the texel for a predetermined period of time after the received request.

26. (new) An address reordering method according to claim 25, wherein the act of saving the texel comprises storing the texel in the texture memory.

27. (new) An address reordering method according to claim 25, further comprising receiving a second request for the texel and reusing the stored texel, responsive to the second request.